HTTPS Traffic:

HTTPS is secure version of HTTP, and the refers to SSL/TLS (Secure Socket Layer/Transport Layer Security).

As the HTTPS is also over TCP, its connection starts with 3 way handshake but it's a bit complex:

- Both the client and the server need to agree on the Protocol Version
- Both the client and the server need to select cryptographic algorithms.
- And some times based on the configuration the server and the client authenticate to each other, like if its locally then both probably need to authenticate to each other to make sure the right people connect and to the right it server people connect
- It uses public key cryptography.

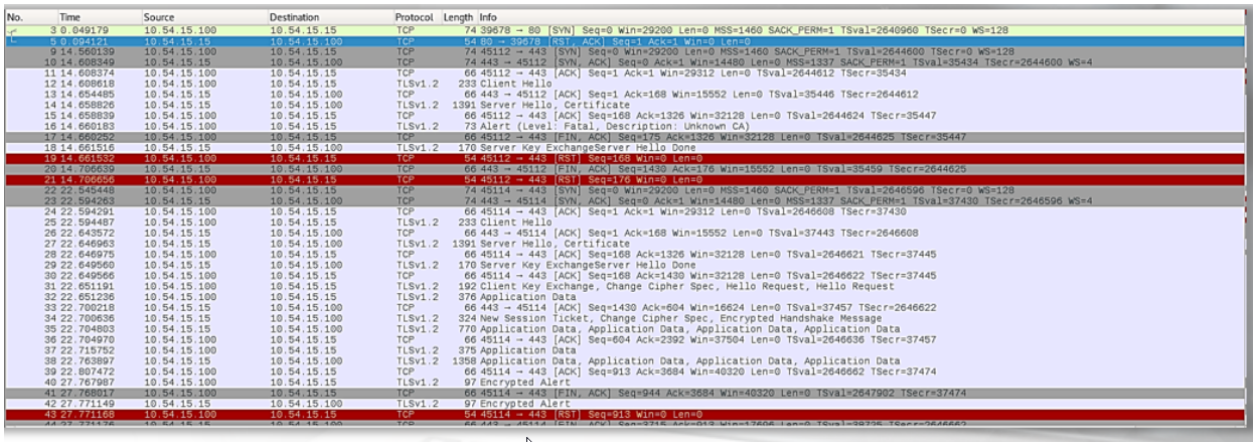
Some facts to distinguish Normal and suspicious HTTP traffic:

Normal HTTPS Traffic	Suspicious HTTPS Traffic
Port 443, TCP Port 8443, TCP (used as alternate)	Malicious binaries (backdoors), scripts, web shells, etc. will use this port because typically in all corporate environments the port is open.
Encrypted traffic	If the traffic is not encrypted and Secure Sockets Layer packet details are empty within packet details, then that will fall under suspicious.
Web server typically in FQDN format.	Server will point to an IP address instead of FQDN format.

HTTPs standard port is 443 TCP and its alternative port is 8443 TCP, if we see it on any other port then it can be suspicious. And if we see the traffic is not encrypted then its also suspicious and the SSL details should not be empty, and if we see its not communicating to FQDN but an IP, then its also suspicious. And this port can be used by attacker for bad ways, as we know the 443 port is open in corporate environments so it can be targeted often by attacers.

Normal HTTPS traffic:

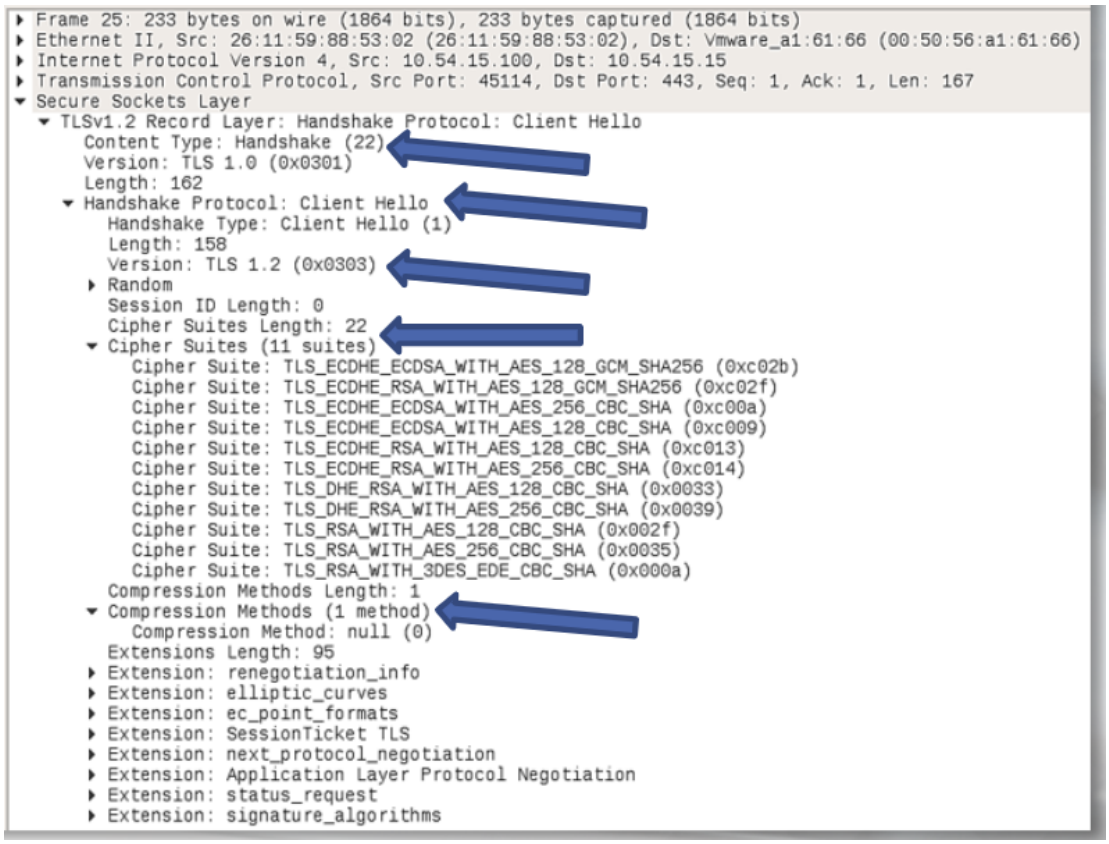
One thing to note is that the HTTPS is encrypted that mean we cant see what is in the traffic.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.049179	10.54.15.100	10.54.15.15	TCP	74	39678 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2640960 TSecr=0 WS=128
5	0.094121	10.54.15.15	10.54.15.100	TCP	54	80 → 39678 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	14.560139	10.54.15.100	10.54.15.15	TCP	74	45112 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2644600 TSecr=0 WS=128
10	14.608349	10.54.15.15	10.54.15.100	TCP	74	443 → 45112 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=35434 TSecr=2644600 WS=4
11	14.608374	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2644612 TSecr=35434
12	14.608618	10.54.15.100	10.54.15.15	TLSv1.2	233	Client Hello
13	14.654485	10.54.15.15	10.54.15.100	TCP	66	443 → 45112 [ACK] Seq=1 Ack=168 Win=15552 Len=0 TSval=35446 TSecr=2644612
14	14.658826	10.54.15.15	10.54.15.100	TLSv1.2	1391	Server Hello, Certificate
15	14.658839	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [ACK] Seq=168 Ack=1326 Win=32128 Len=0 TSval=2644624 TSecr=35447
16	14.660183	10.54.15.100	10.54.15.15	TLSv1.2	73	Alert (Level: Fatal, Description: Unknown CA)
17	14.660252	10.54.15.100	10.54.15.15	TCP	66	45112 → 443 [FIN, ACK] Seq=175 Ack=1326 Win=32128 Len=0 TSval=2644625 TSecr=35447
18	14.661516	10.54.15.15	10.54.15.100	TLSv1.2	170	Server Key ExchangeServer Hello Done
19	14.661532	10.54.15.100	10.54.15.15	TCP	54	45112 → 443 [RST] Seq=168 Win=0 Len=0
20	14.706539	10.54.15.15	10.54.15.100	TCP	66	443 → 45112 [FIN, ACK] Seq=1430 Ack=176 Win=15552 Len=0 TSval=35459 TSecr=2644625
21	14.706656	10.54.15.100	10.54.15.15	TCP	54	45112 → 443 [RST] Seq=176 Win=0 Len=0
22	22.545448	10.54.15.100	10.54.15.15	TCP	74	45114 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2646596 TSecr=0 WS=128
23	22.594263	10.54.15.15	10.54.15.100	TCP	74	443 → 45114 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_PERM=1 TSval=37430 TSecr=2646596 WS=4
24	22.594291	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2646608 TSecr=37430
25	22.594487	10.54.15.100	10.54.15.15	TLSv1.2	233	Client Hello
26	22.643572	10.54.15.15	10.54.15.100	TCP	66	443 → 45114 [ACK] Seq=1 Ack=168 Win=15552 Len=0 TSval=37443 TSecr=2646608
27	22.646963	10.54.15.15	10.54.15.100	TLSv1.2	1391	Server Hello, Certificate
28	22.646975	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=168 Ack=1326 Win=32128 Len=0 TSval=2646621 TSecr=37445
29	22.649560	10.54.15.15	10.54.15.100	TLSv1.2	170	Server Key ExchangeServer Hello Done
30	22.649566	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=168 Ack=1430 Win=32128 Len=0 TSval=2646622 TSecr=37445
31	22.651191	10.54.15.100	10.54.15.15	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
32	22.651236	10.54.15.100	10.54.15.15	TLSv1.2	376	Application Data
33	22.700218	10.54.15.15	10.54.15.100	TCP	66	443 → 45114 [ACK] Seq=1430 Ack=604 Win=16624 Len=0 TSval=37457 TSecr=2646622
34	22.700636	10.54.15.15	10.54.15.100	TLSv1.2	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
35	22.704803	10.54.15.15	10.54.15.100	TLSv1.2	770	Application Data, Application Data, Application Data, Application Data
36	22.704970	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=604 Ack=2392 Win=37504 Len=0 TSval=2646636 TSecr=37457
37	22.715752	10.54.15.100	10.54.15.15	TLSv1.2	375	Application Data
38	22.763897	10.54.15.15	10.54.15.100	TLSv1.2	1358	Application Data, Application Data, Application Data, Application Data
39	22.807472	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [ACK] Seq=913 Ack=3684 Win=40320 Len=0 TSval=2646662 TSecr=37474
40	27.767987	10.54.15.100	10.54.15.15	TLSv1.2	97	Encrypted Alert
41	27.768017	10.54.15.100	10.54.15.15	TCP	66	45114 → 443 [FIN, ACK] Seq=944 Ack=3684 Win=40320 Len=0 TSval=2647902 TSecr=37474
42	27.771149	10.54.15.15	10.54.15.100	TLSv1.2	97	Encrypted Alert
43	27.771168	10.54.15.100	10.54.15.15	TCP	54	45114 → 443 [RST] Seq=913 Win=0 Len=0

By looking at this traffic we cant really know what is going on.

Lets look at one of the packets details:



```
▸ Frame 25: 233 bytes on wire (1864 bits), 233 bytes captured (1864 bits)
▸ Ethernet II, Src: 26:11:59:88:53:02 (26:11:59:88:53:02), Dst: Vmware_a1:61:66 (00:50:56:a1:61:66)
▸ Internet Protocol Version 4, Src: 10.54.15.100, Dst: 10.54.15.15
▸ Transmission Control Protocol, Src Port: 45114, Dst Port: 443, Seq: 1, Ack: 1, Len: 167
▾ Secure Sockets Layer
  ▾ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 162
    ▾ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 158
        Version: TLS 1.2 (0x0303)
      ▸ Random
        Session ID Length: 0
        Cipher Suites Length: 22
      ▾ Cipher Suites (11 suites)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
          Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
          Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
          Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
        Compression Methods Length: 1
      ▾ Compression Methods (1 method)
          Compression Method: null (0)
        Extensions Length: 95
      ▸ Extension: renegotiation_info
      ▸ Extension: elliptic_curves
      ▸ Extension: ec_point_formats
      ▸ Extension: SessionTicket TLS
      ▸ Extension: next_protocol_negotiation
      ▸ Extension: Application Layer Protocol Negotiation
      ▸ Extension: status_request
      ▸ Extension: signature_algorithms
```

As we said earlier that the SSL portion of the packet details should not be empty as if its then its suspicious. Here we see the SSL details.

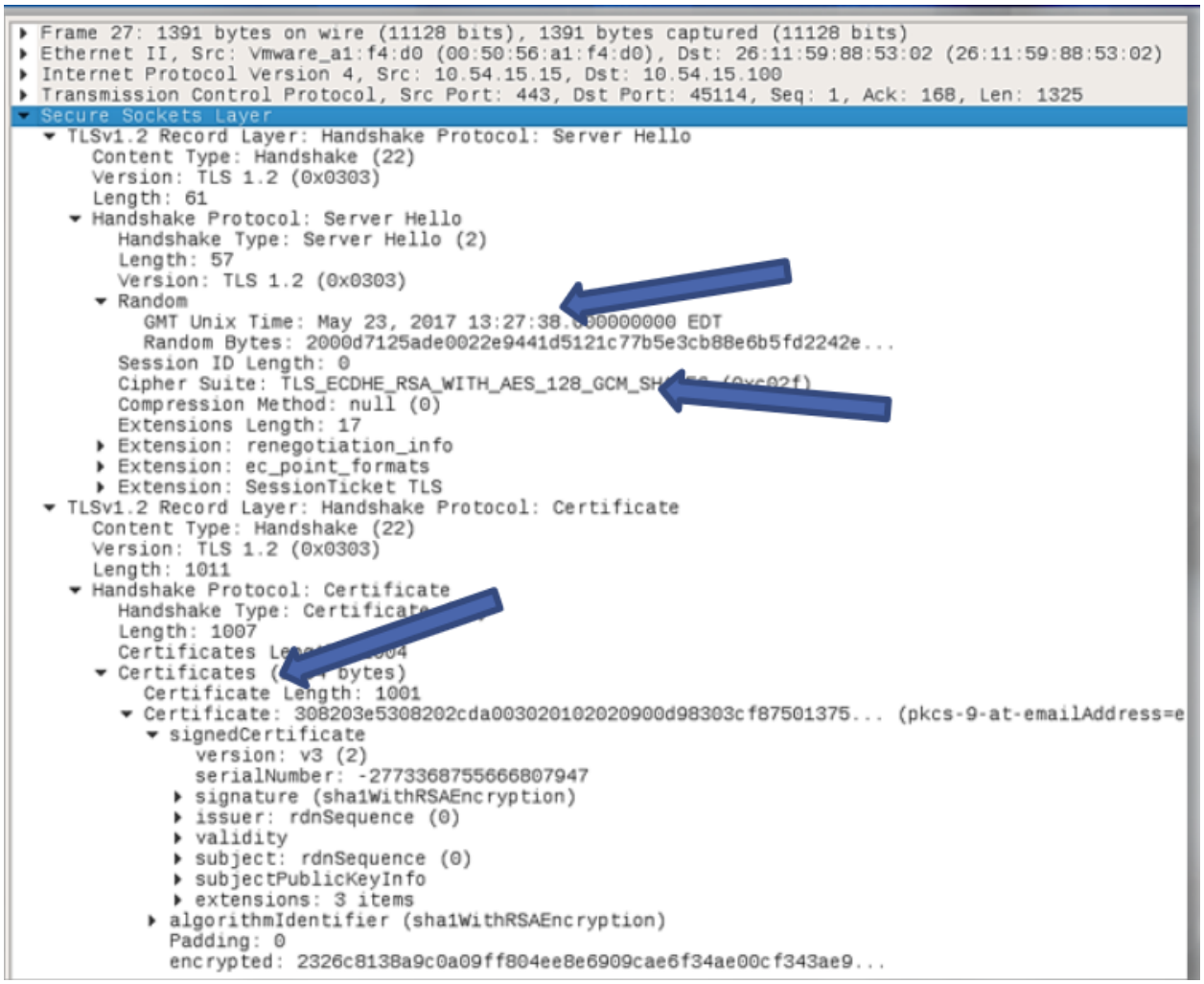
Here we see the client is telling to the server in Client hello, what is available for the client in order to attempt to establish secure connection to server, like the Ciphers which we see it listed 11 of them, in

this packet the Client tell the Server that it can do one of these 11 cipher. As we said the that the TCP 3 way handshake is complex with HTTPS, as it also has the SSL/TLS session establishment as part of the handshake.

We see in the packet:

- Content type: handshake
- Handshake protocol: client Hello
- Version: TLS 1.2
- Cipher Suites (11 suites)
- Compression method (1 method)

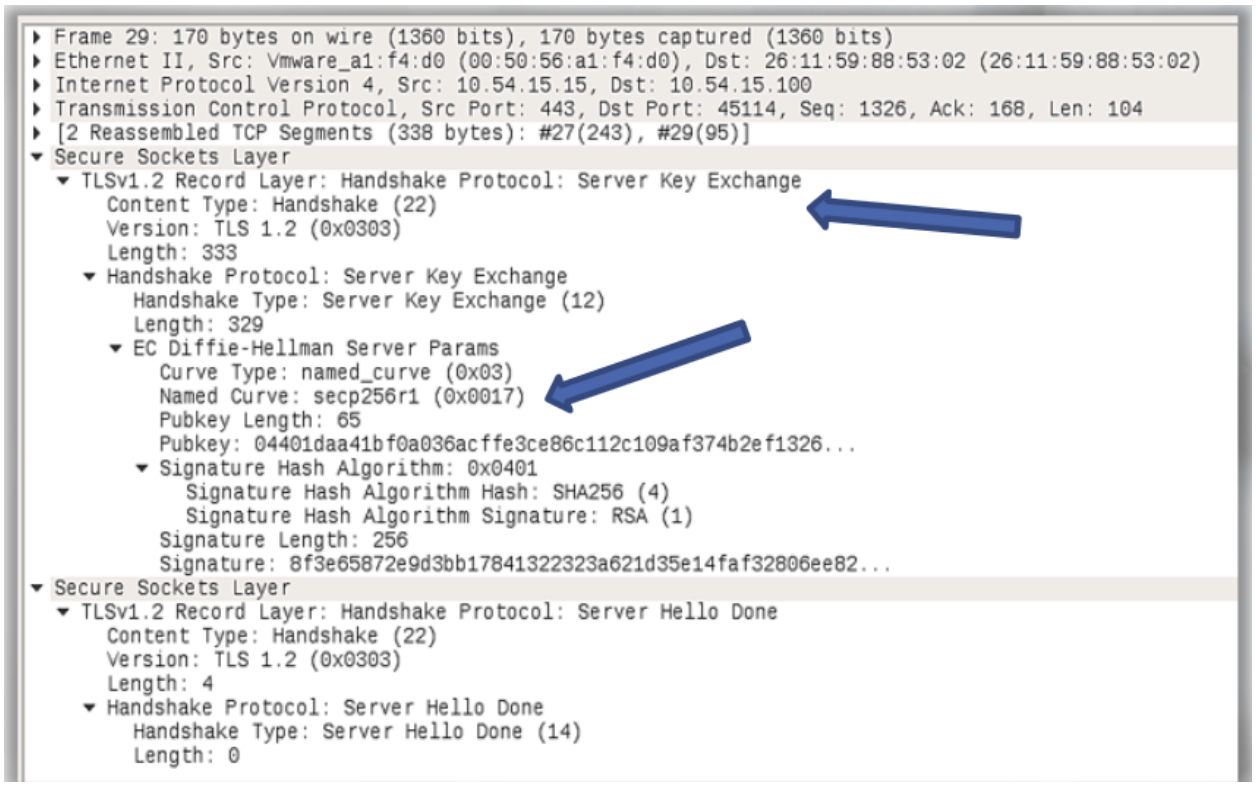
Server Hello packet:



Here this is the response to the Client Hello, and here the server tell about its availability and agreement of encryption, mean in Client Hello the client told the server about the cipher it can use for encryption

and the Server will agree to one of those Ciphers. And here the Server also gave its certificate and other values for the client that it can use to generate a key for the encryption.

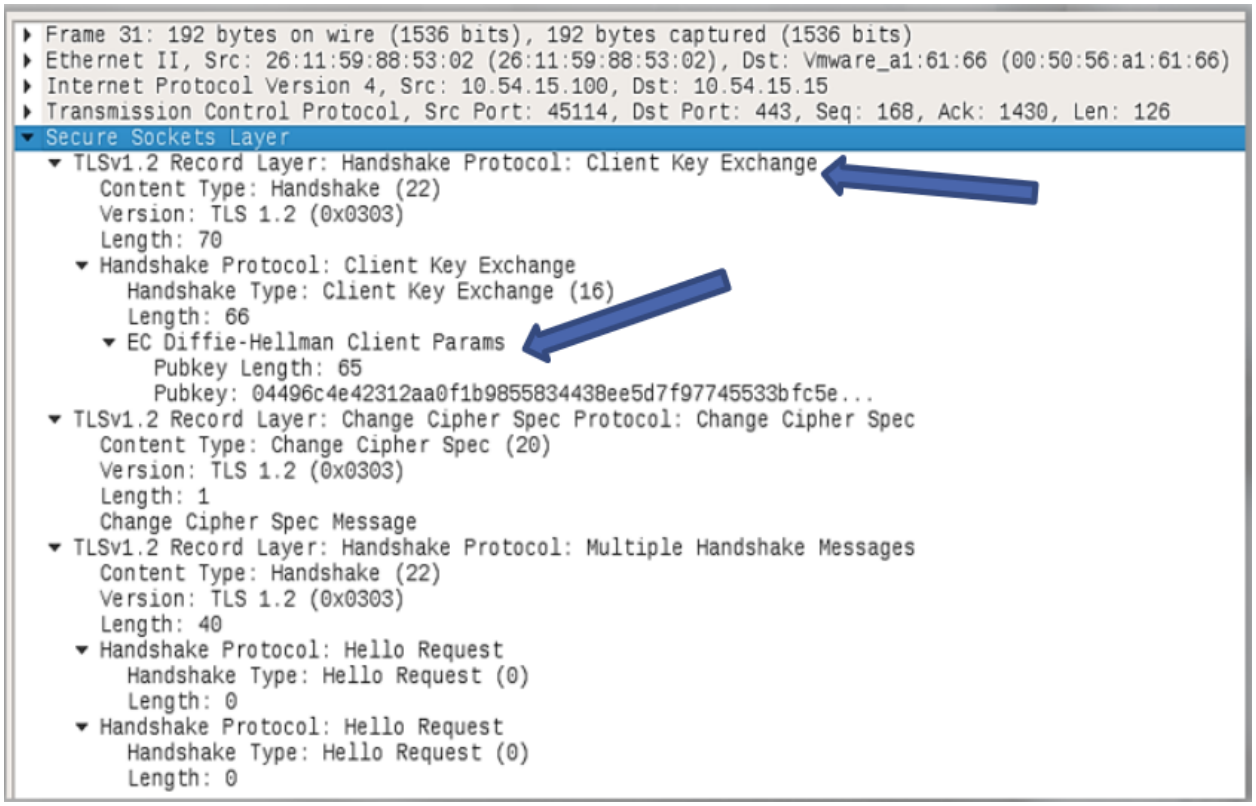
Server Key Exchange:



Then the server will exchange its public key so the client can use it to exchange its symmetric key and then the server will use the client symmetric key to send its symmetric key.

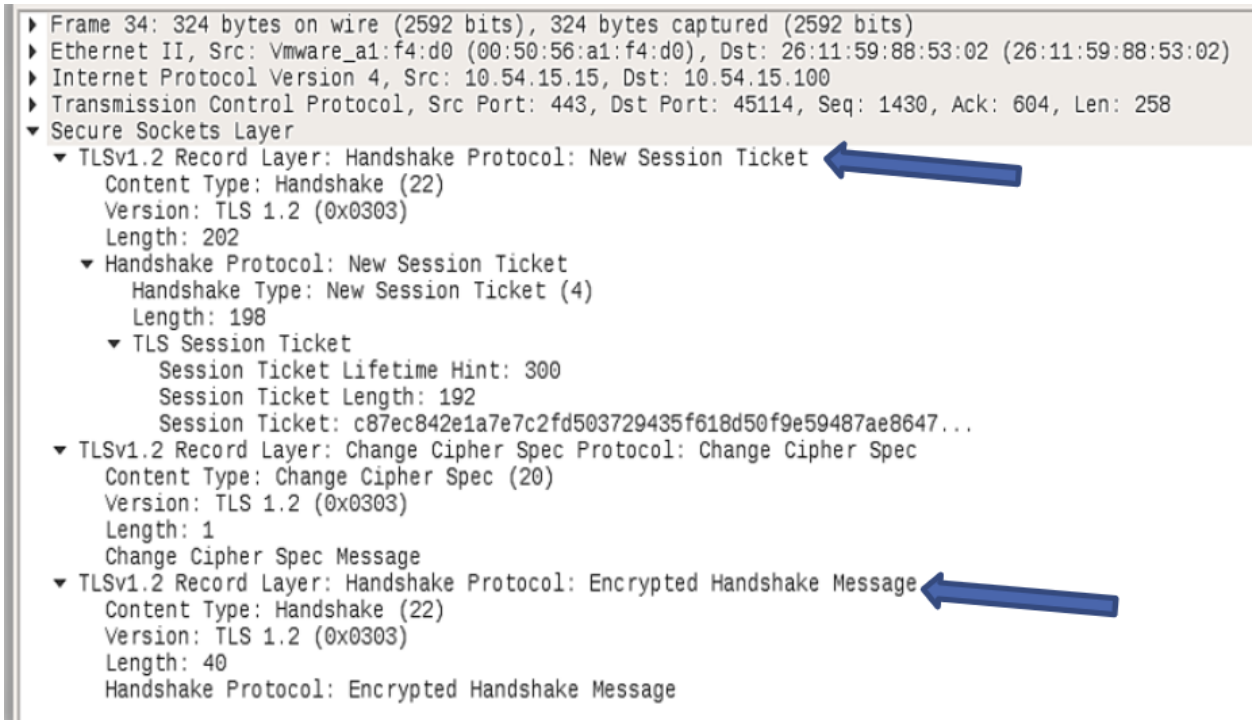
Mean the HTTPS will use the asymmetric and symmetric keys. So the with asymmetric the server and client share its public keys, the server will use the client public key and the client will use the server public key to encrypt the asymmetric keys and sends to each other.

Client Key Exchange:



When the client received the public key of the Server, it will use that key to send its key so the server can use that key to encrypt the traffic.

The last packet of the SSL/TLS handshake:



Summary SSL/TLS Handshake:

The process of key exchange in HTTPS (HTTP Secure) involves several steps to establish a secure communication channel between a client (usually a web browser) and a server. This process is commonly known as the SSL/TLS (Secure Sockets Layer/Transport Layer Security) handshake. TLS is the modern successor to SSL, and the term "TLS handshake" is more accurate for the current technology. Here's a high-level overview of how the TLS handshake works:

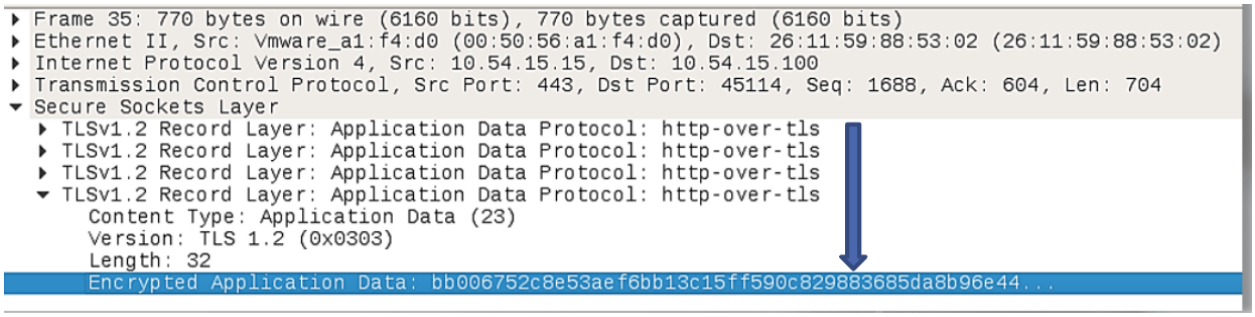
1. **Client Hello:** The TLS handshake begins when the client (web browser) sends a "ClientHello" message to the server. This message includes information about the supported encryption algorithms and other parameters.
2. **Server Hello:** In response to the ClientHello, the server sends a "ServerHello" message. This message contains information about the selected encryption algorithm, along with the server's digital certificate.
3. **Server Certificate:** The server sends its digital certificate to the client. This certificate contains the server's public key and is used to verify the server's identity.
4. **Key Exchange:** The client generates a random "pre-master secret" and encrypts it using the server's public key from the received certificate. This encrypted pre-master secret is sent to the server.

5. **Pre-Master Secret Decryption:** The server decrypts the encrypted pre-master secret using its private key. Both the client and the server now independently have the same pre-master secret.
6. **Session Key Derivation:** Both the client and the server use the pre-master secret to independently derive the same "master secret." This master secret is used to generate symmetric encryption keys for the session.
7. **Finished Messages:** The client and the server exchange "Finished" messages to confirm that the handshake process is complete and that they are ready to begin secure data transmission.
8. **Secure Data Exchange:** Once the handshake is complete, the client and the server can securely exchange data using the symmetric encryption keys derived from the master secret.

Throughout this process, asymmetric encryption (public-key cryptography) is used primarily during the initial steps to securely exchange the pre-master secret, and symmetric encryption (private-key cryptography) is then used for efficient and secure data transmission.

The TLS handshake ensures that the communication between the client and the server is encrypted, authenticated, and secure from eavesdropping and tampering. It establishes a secure channel over which the actual HTTP data (requests and responses) can be transmitted securely.

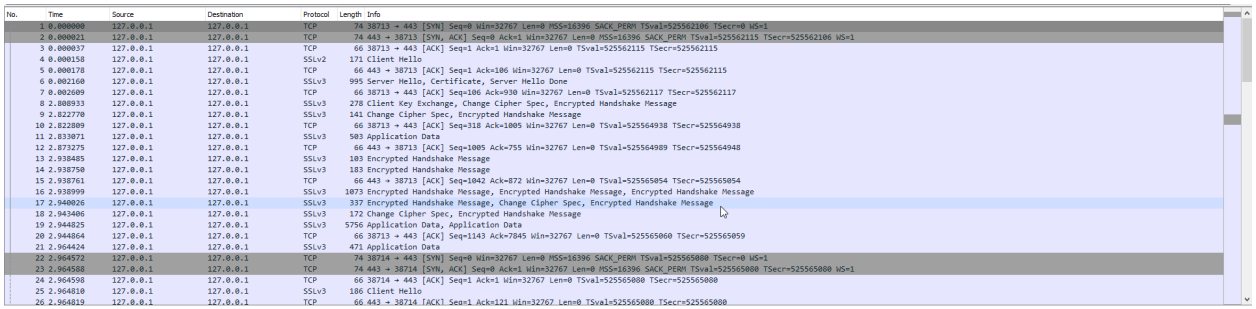
As we said before the HTTPS traffic is encrypted so we cant really look into it and see what data is exchanged.



Here we the data is encrypted:

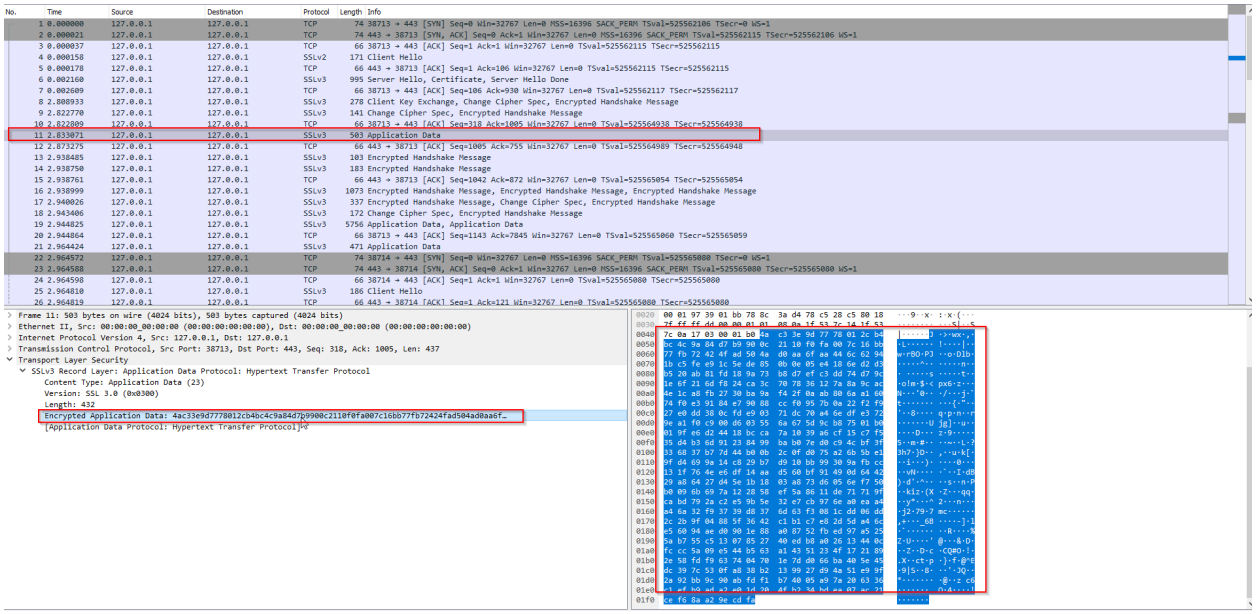
There is ways to decrypt this traffic like if we using Proxy servers where the Proxy server will be making the requests on behalf of the user, so that mean the server is making the request not the user so that way the Proxy server will see the traffic in clear text as the SSL/TSL session is created with the Proxy server not directly with the client.

Lets look at a Packet capture:



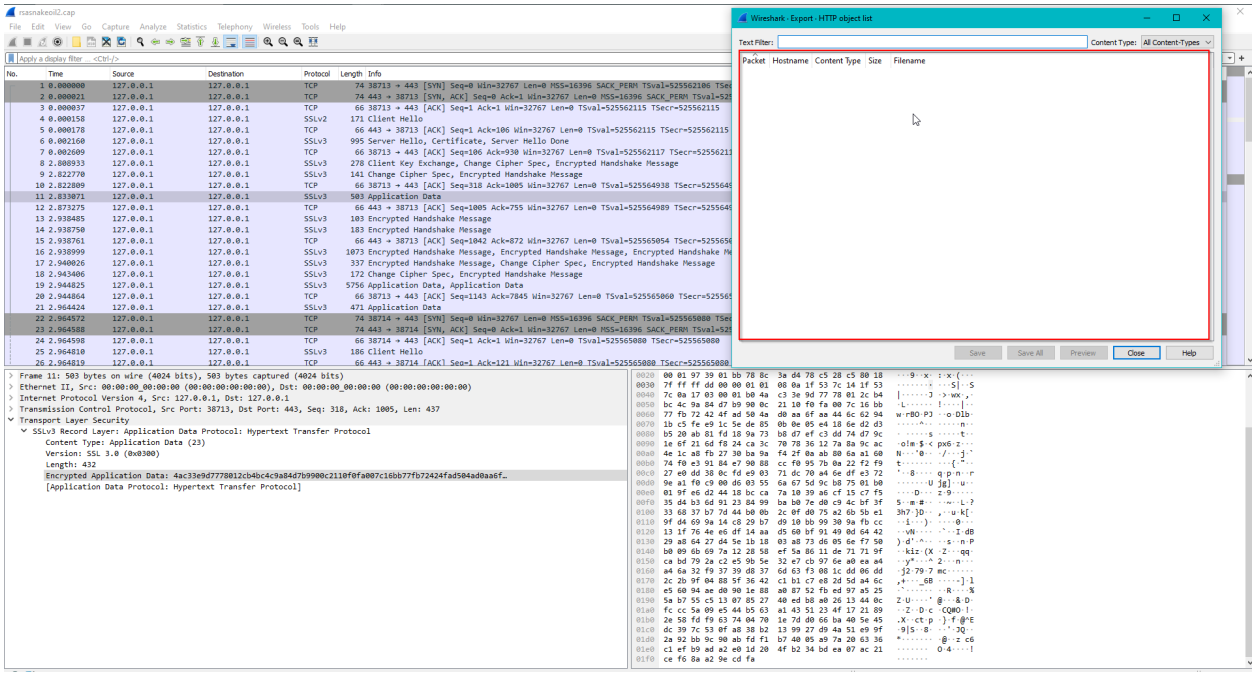
This is packet capture is SSL encrypted, not TLS so its handshake is a bit deferent:

Lets look at packet where its after the handshake where the real data is start to exchange:



Here we clearly see the data is not readable as it should as its encrypted. We will decrypt it but lets look at the export to see if any File like image, binary is available in this traffic, as we know we shouldn't be able to see it as its encrypted but still:

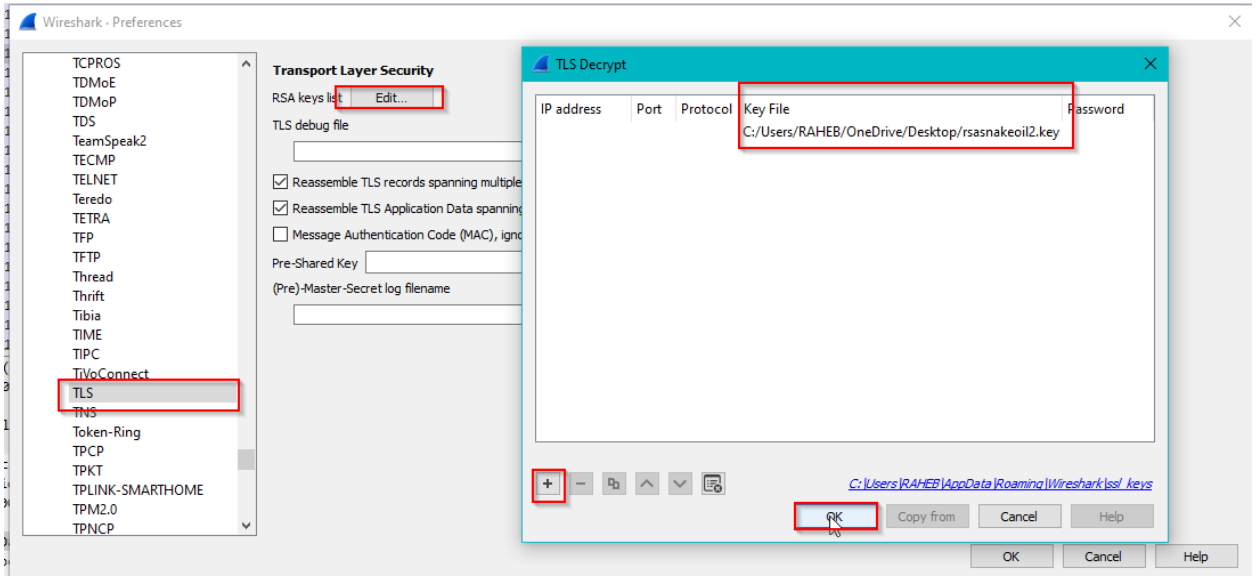
File > Export objects > HTTP



We see nothing as we should not as the traffic is encrypted:

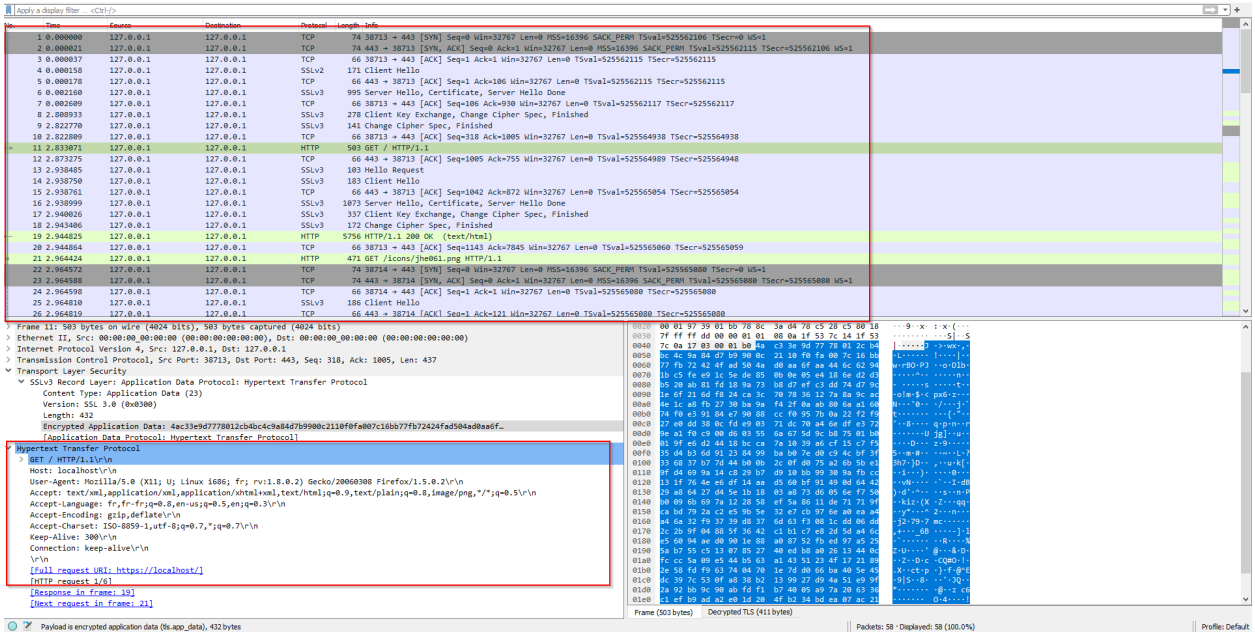
We have the key decrypt this traffic, and to do so we will go to:

Edit > Preference > Protocols > TLS and then select Edit Next to RSA Keys list:



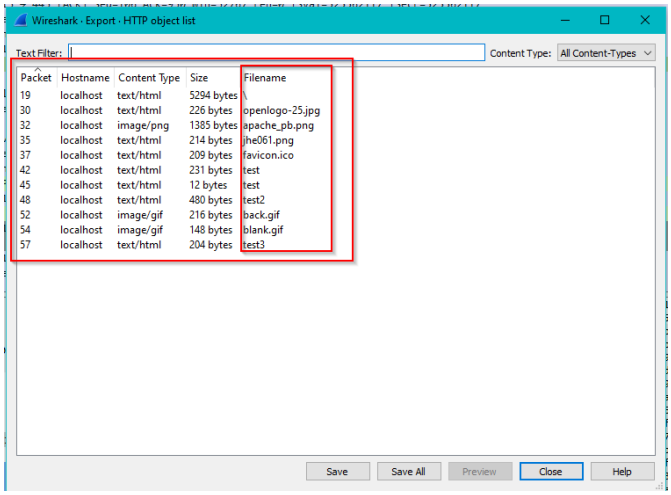
And then we add the new key and select the key file which we downloaded with pcap file and click ok:

The traffic is now decrypted:



We can see the HTTP protocol show up in the Packet details and we see in the packet pane the paths and methods of the HTTP.

Lets now check the Export option and see if there is any files exchanged within this traffic or not:



We do see files here and we see in which packets these were exchanged.

Suspicious HTTP Traffic:

Here we will at the “CrypMIC Ransomware” within the malicious Traffic, we will into how the malware used port 443 to display the ransomware note to the victim.

The CyroMIC is a threat that use custom payload via TCP port 443 to communicate to their C&C servers.

Lets first look at the statistics of the Protocols used in the PCAP: to do this we go to:

Statistics > Protocol Hierarchy:

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	878	100.0	710011	23 k	0	0	0
Ethernet	100.0	878	1.7	12292	413	0	0	0
Internet Protocol Version 4	100.0	878	2.5	17560	590	0	0	0
Transmission Control Protocol	100.0	878	95.6	678479	22 k	676	429856	14 k
Secure Sockets Layer	4.0	35	5.0	35786	1203	35	35786	1203
Hypertext Transfer Protocol	1.3	11	12.2	86590	2912	6	2200	73
Media Type	0.1	1	10.9	77493	2606	1	77675	2612
Line-based text data	0.5	4	1.5	10709	360	4	6562	220
Data	17.8	156	28.8	204260	6869	156	204260	6869

Here we see the 100% of the packets are over TCP but only 11 are HTTP(S). we also see the 35 being the SSL which it can be some of it the SSL handshakes but we know 11 of them are HTTP.

Lets look at the Endpoints: mean all the IPs:

Statistics > Endpoints:

Ethernet · 2 | IPv4 · 5 | IPv6 | TCP · 15 | UDP

Address	Port	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Country	AS Number	City	Latitude	Longitude
51.254.30.225	80	532	445 k	326	431 k	206	14 k	France	AS16276 OVH SAS	—	48.858200	2.338700
83.217.27.178	80	24	2184	7	839	17	1345	Russian Federation	AS200161 DATAPRO Limited Liability Company	—	55.738602	37.606800
85.14.243.9	443	311	257 k	200	250 k	111	6792	Germany	AS24961 myLoc managed IT AG	—	51.299301	9.490900
89.36.89.80	80	11	4558	5	3861	6	697	Romania	AS43938 SC Gateway Telecom SRL	Bucharest, 10	44.700001	26.450001
192.168.4.196	49284	11	4558	6	697	5	3861	—	—	—	—	—
192.168.4.196	49288	6	366	4	246	2	120	—	—	—	—	—
192.168.4.196	49289	18	1818	13	1099	5	719	—	—	—	—	—
192.168.4.196	49293	126	88 k	58	4721	68	84 k	—	—	—	—	—
192.168.4.196	49294	12	726	10	606	2	120	—	—	—	—	—
192.168.4.196	49301	394	356 k	138	8799	256	347 k	—	—	—	—	—
192.168.4.196	49310	8	526	5	352	3	174	—	—	—	—	—
192.168.4.196	49311	13	2441	7	438	6	2003	—	—	—	—	—
192.168.4.196	49312	273	253 k	89	5358	184	248 k	—	—	—	—	—
192.168.4.196	49313	9	600	5	320	4	280	—	—	—	—	—
192.168.4.196	49314	8	498	5	324	3	174	—	—	—	—	—

We see here the traffic is going to some countries like Russia, but here we focus on 443 which its traffic is going to Germany.

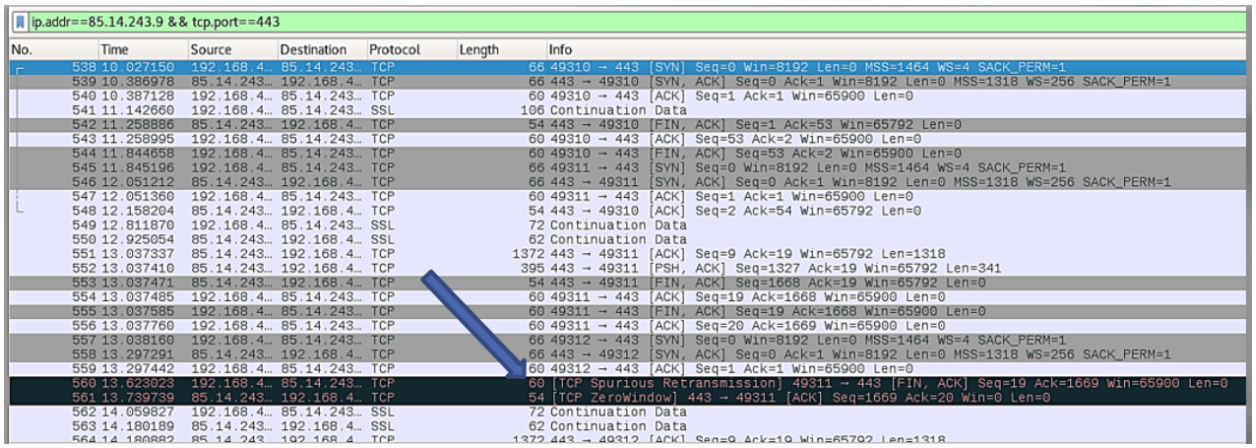
Lets look at the Conversation where we will see which host communicate to IP:

Ethernet · 1 | IPv4 · 4 | IPv6 | TCP · 11 | UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.4.196	49284	89.36.89.80	80	11	4558	6	697	5	3861	0.000000	1.9715	2828	15 k
192.168.4.196	49288	83.217.27.178	80	6	366	4	246	2	120	1.873187	174.1499	11	5
192.168.4.196	49289	83.217.27.178	80	18	1818	13	1099	5	719	1.873282	235.9954	37	24
192.168.4.196	49293	51.254.30.225	80	126	88 k	58	4721	68	84 k	3.462278	221.1622	170	3040
192.168.4.196	49294	51.254.30.225	80	12	726	10	606	2	120	3.462375	232.5032	20	4
192.168.4.196	49301	51.254.30.225	80	394	356 k	138	8799	256	347 k	7.199412	2.5818	27 k	1076 k
192.168.4.196	49310	85.14.243.9	443	8	526	5	352	3	174	10.027150	2.1311	1321	653
192.168.4.196	49311	85.14.243.9	443	13	2441	7	438	6	2003	11.845196	1.8945	1849	8457
192.168.4.196	49312	85.14.243.9	443	273	253 k	89	5358	184	248 k	13.038160	2.5466	16 k	779 k
192.168.4.196	49313	85.14.243.9	443	9	600	5	320	4	280	16.852456	1.3813	1853	1621
192.168.4.196	49314	85.14.243.9	443	8	498	5	324	3	174	84.801036	1.6513	1569	842

Here we see 5 conversation happen over port 443 to the IP 85.14.243.9, and we see in the Endpoints, this IP is from Germany.

Lets add a filter for this 85.14.243.9 IP and Port 443 TCP



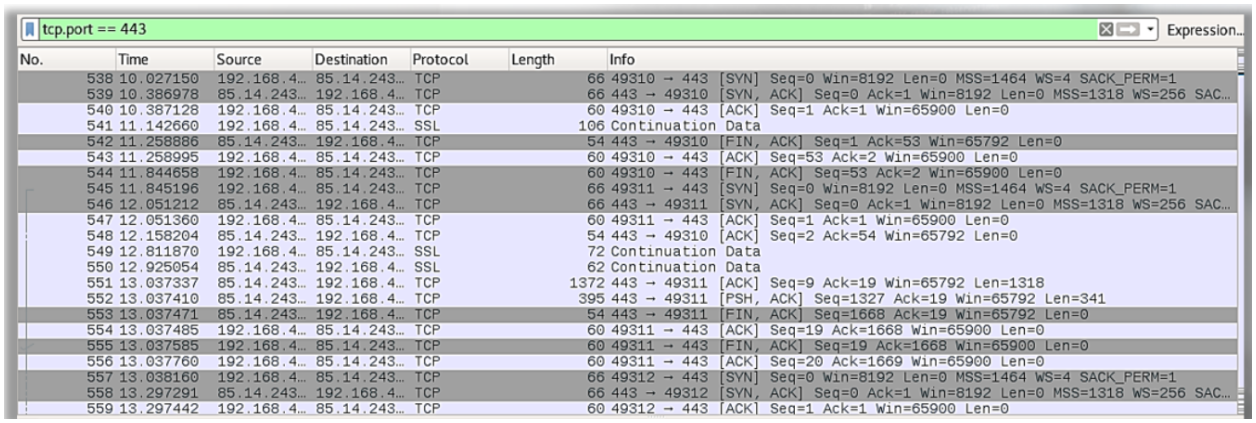
The traffic is encrypted ofc, but we see in packet 560 and 561 that its missing some data, mean we see the TCP Spurious Retransmission which means the host was sending some data and there was an error or the host think like that the receiving host didn't receive the packet so it will send it again.

But then the attacker IP send the TCP ZeroWIndow in packet 561. a TCP Zero Window condition occurs when the receiving device's buffer is completely full, and it cannot accept any more data. It responds to the sender with a TCP segment containing a zero window size to indicate that the sender should pause data transmission.

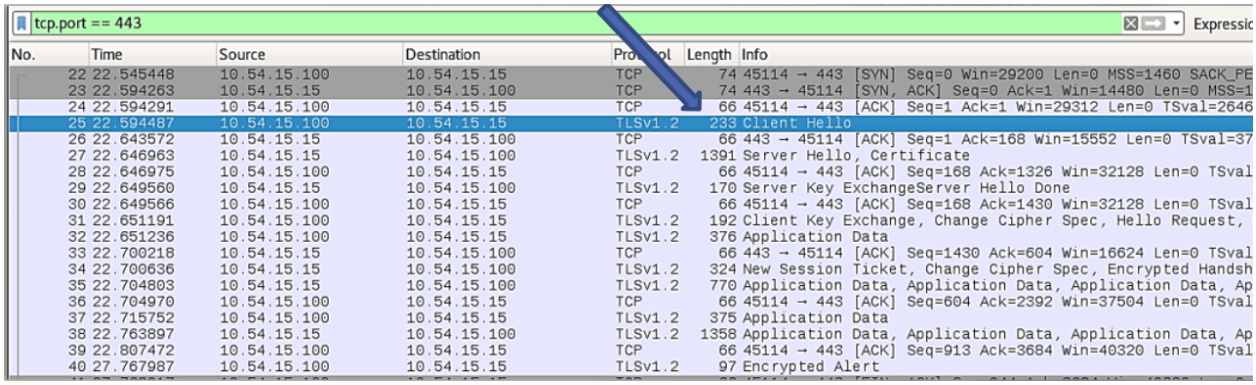
But here if we look at this packet 561 TCP stream we will see clear text traffic which will have the ransom note that was displayed to the victim.

So we know this traffic was sent in 443 and its clear text as we see it.

And also we don’t see any SSL/TLS handshake being happen.



As if there was we would see “Client Hello” like the screen shot bellow.



One another thing to note about SSL is that the SSL Part of the Packet should not be empty: